MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo 1136 August 1989

Computer Perception of Three-Dimensional Objects

Thomas Marill

Abstract

We first pose the following problem: To develop a program which takes line-drawings as input and constructs three-dimensional objects as output; such that the output objects are the same as the ones we see when we look at the input line-drawing. We then introduce the principle of minimum standard-deviation of angles (MSDA) and discuss a program based on MSDA. We present the results of testing this program with a variety of line-drawings and show that the program constitutes a solution to the stated problem over the range of line-drawings tested. Finally, we relate this work to its historical antecedents in the psychological and computer-vision literature.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institue of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defence under Office of Naval Research contract N00014-85-K-0124.

1. Problem Definition.

1.1 Discussion.

How does a flat image on the retina lead to the perception of three-dimensional objects? This is one of the central mysteries of vision, and it is the problem that concerns us here.

We do not propose, however, to deal with retinal images in their full generality (high-resolution, time-varying, in color). Instead, we note that the phenomenon of three-dimensional perception holds for a class of very simple images, namely line-drawings ("a noble class of pictures," as Sugihara [1] calls them), and we will focus on this class.

In fact, some studies (e.g., Roberts [2]) that start with inputs more complex than line-drawings (grey-scale images) reduce these inputs to line-drawings in the first stage of processing and then proceed to operate on these line-drawings.

To say that we wish to study three-dimensional perception in the context of line-drawings does not constitute a problem definition, however. What exactly is the problem to be solved?

We will take the position that what we seek is a program which, given a line-drawing as input, constructs as output the same three-dimensional objects (if any) as the ones we see when we look at the drawing.

For example, given the line-drawing in cell A of Figure 1, we would like our program to construct a three-dimensional staircase object like the one we see; such that after being rotated 20 degrees, the object produces the image in cell B of Figure 1. (We present below a program that actually performs this task.)

This formulation places the problem in the province of computer vision (we seek a machine implementation) but also puts the issue on a psychological basis (the criterion of correctness is agreement with human vision). It follows that if we achieve a solution to this problem, we will not be able to prove mathematically that the solution is correct. (Only mathematical theorems are mathematically provable, and our desired solution, as we have just formulated it, does not lend itself to being expressed as a theorem.) We may also not be able to say *why* the solution works.

For this and other reasons many authors formulate the problem differently.

According to one school of thought, the problem is that of "recovering" the three-dimensional objects that caused the image. Thus Charniak & McDermott [3]

write: "Unlike many problems in AI, the vision problem may be stated with reasonable precision: Given a two-dimensional image, infer the objects that produced it." The solution to this "recovery" problem is often felt to be a kind of inverse-optical process.

There are several reasons we do not follow this recovery formulation. To begin with, the problem so stated (at least as regards line-drawings and probably also as regards any other form of visual input) is insoluble. Given any line-drawing there is an infinite set of three-dimensional objects whose projection (either perspective or orthographic) is identical to the drawing, and there is no way to know which member of this set caused the image.

What is done, therefore, by writers who subscribe to the recovery formulation, is to impose additional constraints until the problem definition becomes narrow enough to yield results. For example, Huffman [4] and Clowes [5] investigate line-drawings that are projections of objects in the "trihedral world" (where exactly three planar surfaces meet at every corner). Mackworth [6] deals with line-drawings that are correct orthographic projections of opaque polyhedra. Kanade [7] discusses line-drawings that are projections of "origami world" objects. These authors are able to obtain mathematical results in their restricted domains. Even in these domains, however, the results afford only a partial solution, and the problem can hardly be considered to have been solved. Most line-drawings, furthermore, do not fall within these restricted domains.

Furthermore, even if we could somehow recover the three-dimensional object that caused the image, it would not necessarily help: the object recovered might be different from the object which is perceived when we look at the image. In such a case the recovered object might well be considered, on intuitive grounds, to constitute an unsatisfactory solution to the problem.

For example, consider a three-dimensional wire-frame object (call it OBJECT-X) defined by four points and six lines. The points are given by the following Cartesian coordinates.[1]

Points:

[1] Our notation for lists of numbers follows the LISP convention: lists are surrounded by parentheses and the elements of the list are separated by spaces (no commas). Thus the Cartesian coordinates of the point (2.3, 4.5) will be written as (2.3 4.5) and a list consisting of two such points will be written ((2.3 4.5)(2.3 4.5)). This allows lists in the printed text to agree exactly with inputs to and outputs from our program. In the case of mathematical formulas we follow the usual mathematical conventions.

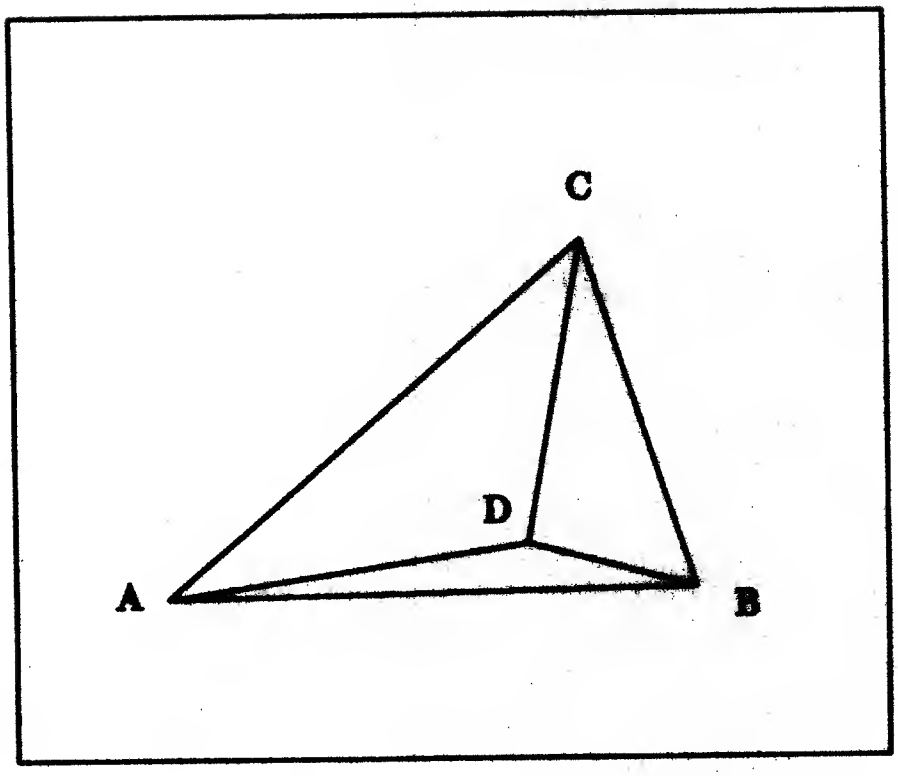

Figure 2.

Orthographic projection of OBJECT-X

A: (-4.33 -1.56 10.0)

B: (1.79 -1.53 200.0)

C: (0.54 2.54 10.0)

D: (-0.14 -0.98 20.0)

The lines are: AB, AC, AD, BC, BD, CD.

The image of OBJECT-X (orthographic projection) is shown in Figure 2.

When we look at Figure 2, we see a tetrahedron with lines AB and AC roughly equal in length. However, in OBJECT-X, line AB is actually 30 times as long as line AC. Even if we were able correctly to recover OBJECT-X from the drawing, we might object to the answer, since the recovered object does not agree with what we plainly see. And it can reasonably be argued that when the object that caused the image is different from the object that we see, it is the latter which represents the desired solution. (If we did not take such a position, our robot systems might end up seeing a world quite different from the world that we see).

So far we have discussed the "psychological" and the "recovery" schools of thought. There is yet another school that differs from both of these. It is the one embodied in the definition of vision given by Ballard & Brown [8]: "Visual perception is the relation of visual input to previously existing *models* of the world." [Ballard & Brown's emphasis].

This formulation leads to a well-defined, and, in principle, soluble, problem: given a set of three-dimensional models and transformations on these models, to find those particular models and transformations whose projection most closely approximates the given image.

Many researchers have worked on "model-based vision." For example, Lowe [9] derived an iterative solution for the problem as formulated above and demonstrated the applicability of this solution to simple polyhedral scenes.

It is clear, however, that the formulation in terms of models also leads to cases in which what the system sees is different from what the human sees. In such a case, as already discussed, one tends to prefer the human solution.

As a problem definition, the model-based approach has one further difficulty. While formulating the vision problem in a reasonable way, it does so at the expense of introducing a new problem that may be even harder to solve than the original one, namely the problem of model acquisition (learning). As Ballard & Brown [8] candidly put it, "Learning is missing from the list above [of topics to be discussed].

Disappointing as it is, at this writing the problem of learning is so difficult that we can say very little about it in the domain of vision."

In summary, then, we have discussed the question of problem definition, and we have described three schools of thought. The first of these holds that the problem is to produce the same solution as the human vision system. The second holds that the task is to recover the object that caused the image. And the third holds that, given a set of models and transformations, the problem is to find those models and transformations that project to the image with the least error. We have given reasons why we subscribe to the first of these.

1.2 Representation of Line-Drawings and Objects.

To be more precise about problem definition we need a formalism for the representation of line-drawings and objects. The representation we will use for line-drawings is illustrated in the following example.

LINE-DRAWING-Y

POINTS: ((3.00 7.15) (2.39 -6.27) (5.00 -9.27) (6.29 -3.0))

LINES: ((0 1) (1 2) (3 0))

This is interpreted to mean that LINE-DRAWING-Y consists of four points and three lines. The four points are expressed as *(x,y)* Cartesian coordinates. The lines are expressed as pairs *(i,j)*, meaning that the i-th point is connected by a straight-line segment to the j-th point ($i, j = 0, 1, 2...$). Thus the first line in LINE-DRAWING-Y runs from point 0 (3.00 7.15) to point 1 (2.39 -6.27).

It should be emphasized that a line-drawing, as defined, is an arbitrary collection of points and lines. We do not restrict attention to line-drawings that are images of certain classes of objects. Our line-drawings, in fact, are not considered to be images *of* anything. They are simply viewed as uninterpreted inputs to the human vision system and to our program.

To each line-drawing expressed in the ("internal") representation given above, there is a unique "external" form: the actual lines and points drawn on a sheet of

paper.[2] We use this "external" form in our illustrations.[3]

The way in which we will represent three-dimensional objects is identical to the way we represent line-drawings except that points in objects have three coordinates *(x,y,z)* instead of two. An object so represented agrees with our intuition of a "wire frame".[4]

Thus, OBJECT-X discussed above is formally represented as follows:

OBJECT-X

POINTS: ((-4.33 -1.56 10.0)(1.79 -1.53 200.0)(0.54 2.54 10.0)(-0.14 -0.98 20.0))

LINES: ((0 1) (0 2) (0 3) (1 2) (1 3) (2 3))

1.3 Problem definition.

In summary, we define our problem to be the following: To develop a program which takes line-drawings as inputs and generates objects as outputs, where the line-drawings and objects are represented as specified above; such that, for a given line-drawing, the ouput agrees with the object that a person sees when he looks at the line-drawing.

---

[2]The converse is not generally true. A given drawing on a sheet of paper may correspond to several (internal) representations. If we start with an externally represented drawing, an interpretive step may be needed to generate the internal representation.

[3]In these illustrations the coordinate axes are understood to be parallel to the edes of the page; the scale and the location of the origin are not significant.

[4]It should be understood that an object so represented is explicitly in a particular location in space. If we were to apply a rigid-body transformation to the points, the "object" would change (even though the lengths of the lines and the magnitude of the angles between the lines would remain invariant). It would be more correct to call the entity represented an "object-in-a-particular-location", though this would be too awkward to be practical.

2. Principles of Operation.

In the present section we discuss the operation of a program (called the CONSTRUCT program) that takes line-drawings as input and constructs objects as output. Examples of the operation of CONSTRUCT are presented in Section 3.

2.1 The Concept of Extension.

We will be using the concept of the extension of a line-drawing. By the *orthographic extension* of a line-drawing L we mean the set E of objects (the objects being represented as discussed above) whose orthographic projection is equal to L. (Similarly, by the *perspective* extension of a line-drawing L we mean the set E of objects whose perspective projection is equal to L.) The extension of a line-drawing is a set with an infinite number of members.

The concept of extension is useful in discussing the construction of objects from line-drawings and other images.

2.2 The MSDA Principle.

The basic principle underlying the operation of the program is extremely simple: it is to minimize the standard-deviation of the angles in the constructed object. We call this the MSDA principle.

Conceptually, we can think of the process as follows. Consider the orthographic extension of the input line-drawing. For each object in this set determine all of its (three-dimensional) angles. Attach to each object a figure-of-merit: the standard-deviation of its angles. Finally, pick that object for which the figure-of-merit is minimized. (If there is more than one object with minimum figure-of-merit in the extension, pick any one of them.) This is the output.

2.3 Angles

It is necessary to look more closely at what we mean by *angle*. Consider a point P in an object (the object being represented as discussed above). P will be the terminus of n lines ($n = 0, 1, ...$). We form all pairs of lines at P and compute an angle magnitude for each pair. Thus, if there are five lines at P, there will be ten angles. If $n < 2$, there are no angles at P.

Suppose l_i and l_j are two lines at P. The magnitude of the angle between l_i and l_j is obtained by first determining the two three-dimensional unit vectors $\mathbf{u}_i$ and $\mathbf{u}_j$, parallel to l_i and l_j, and then computing the angle α_{ij} by the formula

$$\alpha_{ij} = \cos^{-1}(\mathbf{u}_i \cdot \mathbf{u}_j)$$

2.4 Representing Objects in the Extension of a Line-Drawing

Each object in the extension of the input line-drawing is uniquely determined by an n-dimensional vector of numbers $\mathbf{z} : (z_0, ..., z_i, ..., z_{n-1})$, where z_i is the z-coordinate of the i-th point in the object and n is the number of points in the object.

Given the object, we can obviously write $\mathbf{z}$. We can also go the other way: given $\mathbf{z}$ (and the line-drawing) we can construct the object as follows. To each point (x_i, y_i) in the line-drawing, append the i-th element of $\mathbf{z}$, z_i, as the z-coordinate, thereby forming (x_i, y_i, z_i). The line-list remains unchanged. The result will be that object in the extension of the line-drawing that corresponds to $\mathbf{z}$. (This construction follows from the definition of orthographic projection.)[5]

2.5 Hill-Climbing

Thus we see that if we are given a line-drawing L, we can represent each object in the extension of L by a vector $\mathbf{z}$. We can therefore compute the standard-deviation of the angles (SDA) of the object for each vector $\mathbf{z}$.

What we seek is the object with minimum SDA. We tackle this problem with a hill-climbing search over the space of $\mathbf{z}$. Such a search represents a heuristic

[5]It is also true that each object in the *perspective* extension of the input line-drawing is uniquely determined by an n-dimensional vector $\mathbf{z}$. However, the mapping from vector to object is slightly more complicated.

approach to the problem; it is not guaranteed to find a global minimum, but may return an SDA which is only locally minimized. (Furthermore, if there are actually several vectors in the space with minimum SDA, the procedure will find one at most.) Thus, hill-climbing is not infallible. (For a discussion of hill-climbing and its hazard, see Winston [10].)

The input line-drawing is equivalent to the object corresponding to the zero vector **0**; that is, the line-drawing can be thought of as a flat object lying in the x-y plane and thereby having all z-coordinates equal to zero. The search starts here; i.e., the original value of the *current vector* is equal to **0**.

At each stage of the search, the SDA of the current vector will have been computed. The program now looks at the SDA of all the children of the current vector. These children are all of the vectors one step-size away from the current vector. If s is the step-size, and if the current vector is $\mathbf{z} : (z_0, z_1, ..., z_{n-1})$, then the children of the current vector are:

$$(x_0 + s, x_1, ..., x_{n-1})$$
$$(x_0 - s, x_1, ..., x_{n-1})$$
$$(x_0, x_1 + s, ..., x_{n-1})$$
$$(x_0, x_1 - s, ..., x_{n-1})$$
$$\ldots$$
$$(x_0, x_1, ..., x_{n-1} + s)$$
$$(x_0, x_1, ..., x_{n-1} - s)$$

Each of these $2n$ children is assigned an SDA, and the child with minimum SDA is selected as the new current vector (if there is more than one, the first of these is selected). The process then repeats, until no further improvement in SDA is obtained. The vector with the smallest SDA of those inspected is the result of the process.

The above constitutes one "round" of hill-climbing. In the CONSTRUCT program, there are three rounds, with step-size s_1, s_2, and s_3, where $s_1 > s_2 > s_3$. Each round begins its search with the result of the preceding round.

The values of the s_i have been determined experimentally, and are compiled into the CONSTRUCT program to be 1, 0.5, and 0.1 respectively.

2.6 Performance

The hill-climbing procedure tends to be slow. The CONSTRUCT program, written in Common LISP and running on a Symbolics computer, took an average of 30 seconds for the first eight examples of Section 3. (The fastest was 5 seconds and the slowest was 57 seconds). The last example (Section 3.9, Figure 11) is somewhat more complex, and took 320 seconds. Clearly, if the MSDA approach is to find acceptance as a model of real-time vision, a faster technique for computing MSDA will have to be found.

3. Examples.

3.1 Method of Presenting Examples. Example A. (Figure 3).

In the present section we discuss our method of presenting examples, and we do so in the context of an actual example (Example A, illustrated in Figure 3).

We start with a line-drawing, which is shown in cell C of Fig 2.

LINE-DRAWING-A

POINTS: ((-2.89 -1.62) (0.57 -1.62) (0.92 2.32) (-2.54 2.32) (-0.92 -2.32) (2.54 -2.32) (2.89 1.62) (-0.57 1.62))

LINES: ((0 1) (1 2) (2 3) (3 0) (4 5) (5 6) (6 7) (7 4) (0 4) (1 5) (2 6) (3 7)))

When LINE-DRAWING-A is given to the CONSTRUCT program as input, the program generates the following object as output:

OUTPUT-OBJECT-A

POINTS: ((-2.89 -1.62 0.0) (0.57 -1.62 2.0) (0.92 2.32 1.4) (-2.54 2.32 -0.6) (-0.92 -2.32 -3.5) (2.54 -2.32 -1.5) (2.89 1.62 -2.1) (-0.57 1.62 -4.1))

LINES: ((0 1) (1 2) (2 3) (3 0) (4 5) (5 6) (6 7) (7 4) (0 4) (1 5) (2 6) (3 7))

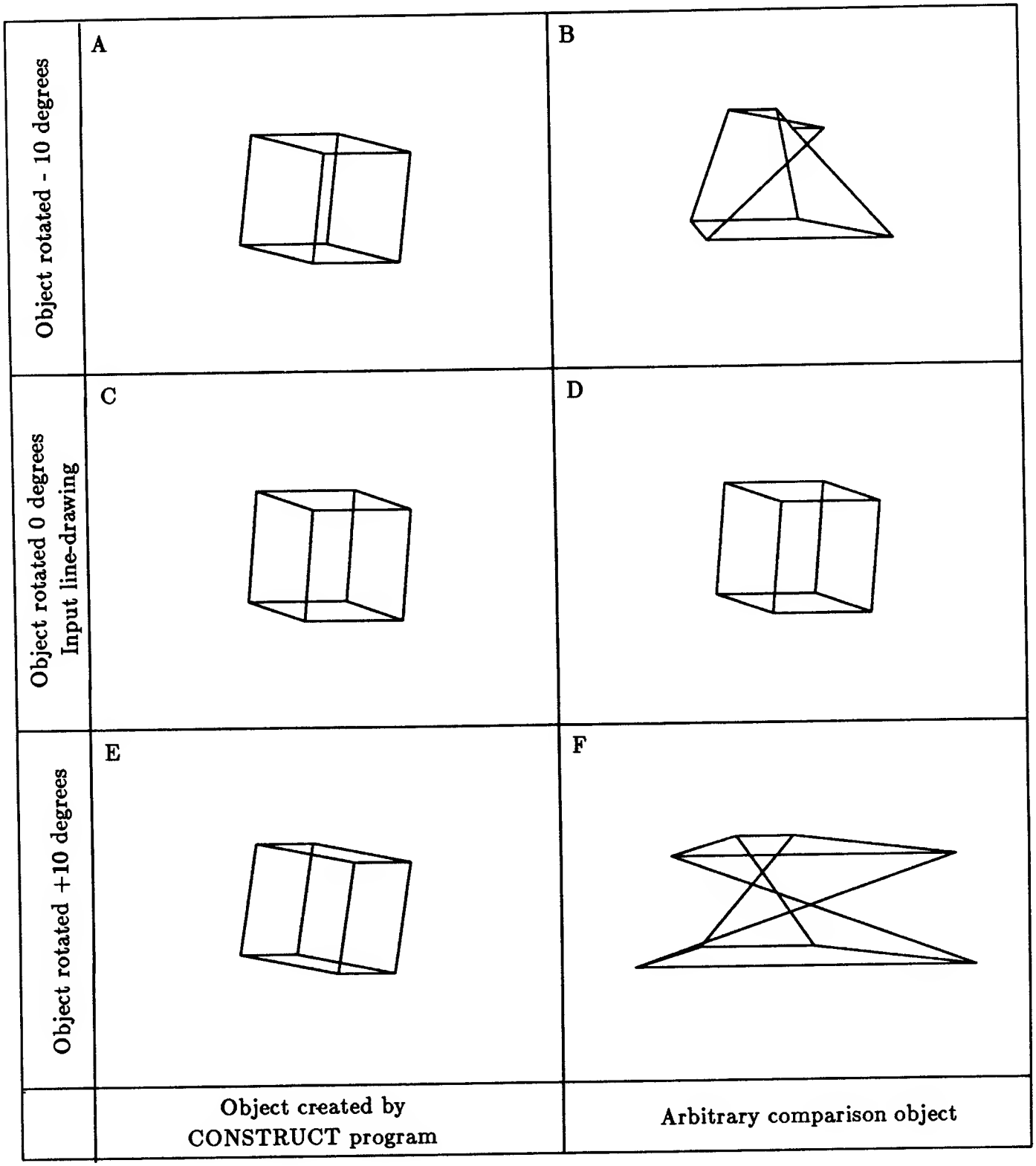

Figure 3.

Example A. See Section 3.1

OUTPUT-OBJECT-A is a cube (to within the precision of the program). A cube is also the object we *see* when we look at the line-drawing in cell C of Figure 3. Hence we say that the program has given the correct answer.

We can also visually determine the correctness of the program output by comparing the three cells in the left-hand column of Figure 3, which represent rotated views of OUTPUT-OBJECT-A.

Cell C in Figure 3 shows the orthographic projection of OUTPUT-OBJECT-A (rotated 0 degrees). This projection is identical to the input line-drawing, a fact that follows directly from the principle of operation of the CONSTRUCT program (see Section 2).

Figure 3 cell A is the orthographic projection of OUTPUT-OBJECT-A after this object has been rotated -10 degrees.[6] Figure 3 cell E is the projection of OUTPUT-OBJECT-A after this object has been rotated +10 degrees.

The object we see when we look at the drawings in cells A and E is the same object as the one we see when we look at C (only rotated). This gives us confidence that the program has produced the correct output object. Experience has shown that an incorrect output is quickly detected visually by comparing C with A and E.

This point is brought home to us when we look at the three cells in the right-hand column of Figure 3. Since the human vision system is so extraordinarily good at forming three-dimensional objects from line-drawings, it is hard for us to keep in mind that there are other objects than the cube that also have C as their image. To remind us of this point, we construct an arbitrary comparison object that projects to the same line-drawing. The object is[7]

COMPARISON-OBJECT-A

POINTS: ((-2.89 -1.62 0.0) (0.57 -1.62 -2.0) (0.92 2.32 4.0) (-2.54 2.32 -6.0) (-0.92 -2.32 8.0) (2.54 -2.32 -10.0) (2.89 1.62 12.0) (-0.57 1.62 -14.0))

LINES: ((0 1) (1 2) (2 3) (3 0) (4 5) (5 6) (6 7) (7 4) (0 4) (1 5) (2 6) (3 7)))

---

[6] Rotation is around a vertical axis through the center of the object. The center is at point (x_c, y_c, z_c), where x_c is the average x-coordinate of the object-points, y_c the average y-coordinate, and z_c the average z-coordinate.

[7] Note that the output object and the comparison object differ only in their z-coordinates. This is a consequence of the fact that they both have the same orthographic projection, namely the input line-drawing.

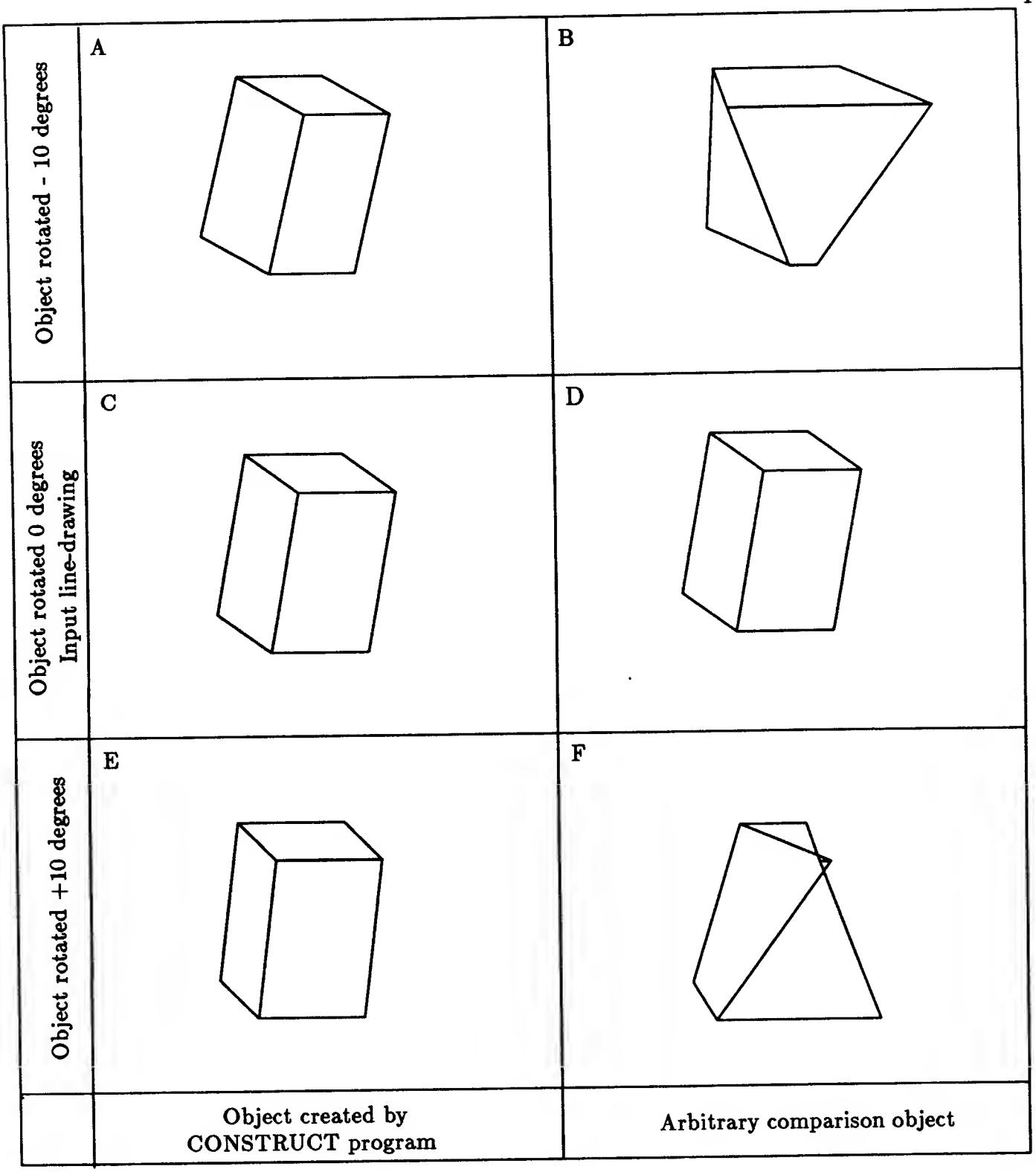

Figure 4.

Example B. See Section 3.2

The comparison object is shown in cell D and its rotations are shown in cells B and F. If the CONSTRUCT program had generated COMPARISON-OBJECT-A instead of OBJECT-A, the error would have been easy to detect visually.[8]

It is sometimes instructive to look at the angles, the standard-deviation of the angles (SDA), and the line-lengths of the constructed object and of the comparison object. These are given in the Appendix for the present example and all subsequent examples.

To review: The three illustrations in the left-hand column (cells A, C, E) represent three views of the three-dimensional object generated by the program. The three illustrations in the right-hand column (cells B, D, F) represent three views of an arbitrary three-dimensional comparison-object. Both objects have the same orthographic projection (before rotation). This projection is shown in cells C and D, and also constitutes the input to the CONSTRUCT program.

The object constructed by the program agrees with what we perceive when we look at the input line-drawing.

3.2 Example B. (Figure 4).

The following input line-drawing is illustrated in Figure 4 cell C.

LINE-DRAWING-B

POINTS: ((-3.43 -1.01) (1.06 4.63) (-2.4 4.63) (-1.55 -2.38) (1.91 -2.38) (2.94 3.26) (-0.53 3.26))

LINES: ((1 2) (2 0) (3 4) (4 5) (5 6) (6 3) (0 3) (1 5) (2 6))

The output from the program is as follows, and is illustrated in the left-hand column of Figure 4.

OUTPUT-OBJECT-B

[8] There is still another technique for examining the output object visually. This technique, which is available to the experimenter but not to the reader of this paper, is to look at a "movie" formed by a rapid sequence of images of the object as the object rotates around its center.

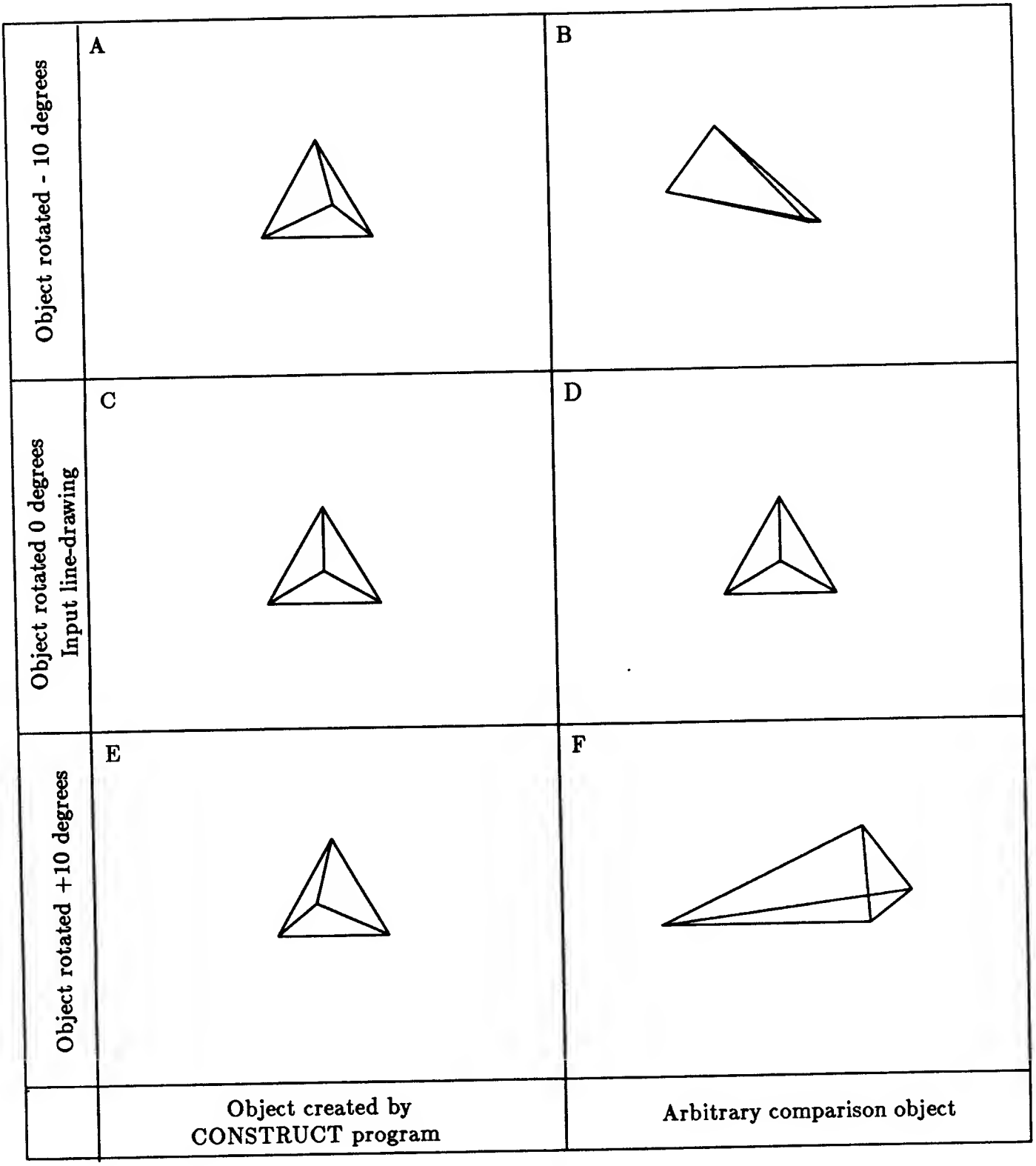

Figure 5.

Example C. See Section 3.3

POINTS: ((-3.43 -1.01 0.0) (1.06 4.63 0.2) (-2.4 4.63 -1.9) (-1.55 -2.38 -3.0) (1.91 -2.38 -0.9) (2.94 3.26 -2.8) (-0.53 3.26 -4.9))

LINES: ((1 2) (2 0) (3 4) (4 5) (5 6) (6 3) (0 3) (1 5) (2 6))

This is a fairly strange object. It can be thought of (among other things) as a rectangular prism with certain lines missing.[9]

We can determine the correctness of the output by looking at the left column in Figure 4 (cells A, C, and E).

The comparison object is:

COMPARISON-OBJECT-B

POINTS: ((-3.43 -1.01 0.0) (1.06 4.63 -2.0) (-2.4 4.63 4.0) (-1.55 -2.38 -6.0) (1.91 -2.38 8.0) (2.94 3.26 -10.0) (-0.53 3.26 12.0))

LINES: ((1 2) (2 0) (3 4) (4 5) (5 6) (6 3) (0 3) (1 5) (2 6))

This comparison object is illustrated in the right-hand column of Figure 4.

3.3 Example C. (Figure 5).

In the two preceding examples the angles of the output object were right angles. In the present example, illustrated in Figure 5, the program constructs an object none of whose angles are right angles.

LINE-DRAWING-C

POINTS: ((0.0 0.0) (4.0 0.0) (2.0 3.46) (2.0 1.15))

LINES: ((0 1) (0 2) (0 3) (1 2) (1 3) (2 3))

The output object is a regular tetrahedron (which is also what we see when we look at Figure 5 cell C:

[9]The lines that are missing are the ones that would fail to project to the image if the object consisted of opaque surfaces. The object may therefore be thought of as having hidden lines eliminated.

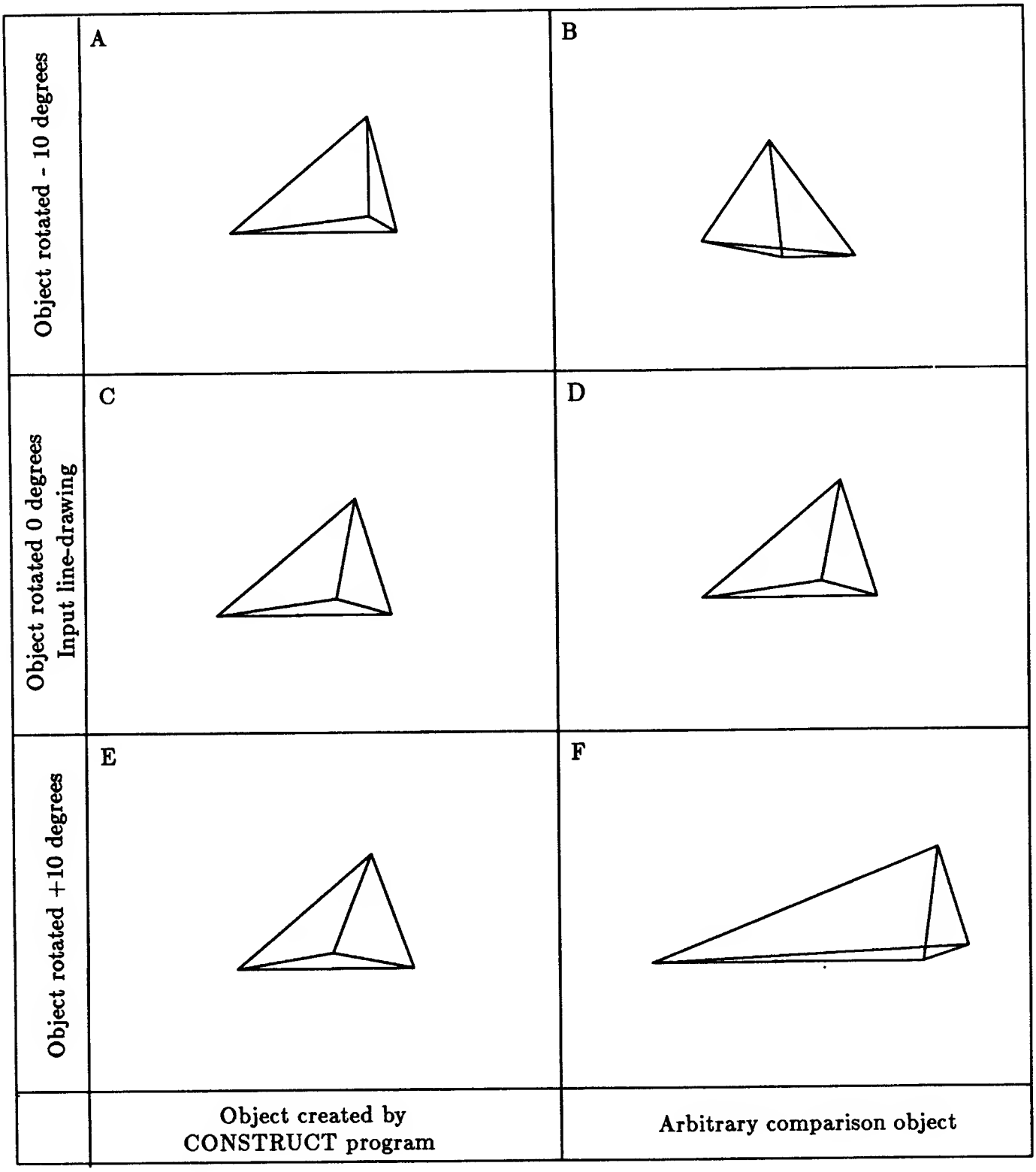

Figure 6.

Example D. See Section 3.4

OUTPUT-OBJECT-C
POINTS: ((0.0 0.0 0.0) (4.0 0.0 0.0) (2.0 3.46 0.0) (2.0 1.15 -3.3))
LINES: ((0 1) (0 2) (0 3) (1 2) (1 3) (2 3))

The comparison-object is:

COMPARISON-OBJECT-C
POINTS: ((0.0 0.0 -10.0) (4.0 0.0 10.0) (2.0 3.46 20.0) (2.0 1.15 30.0))
LINES: ((0 1) (0 2) (0 3) (1 2) (1 3) (2 3))

3.4 Example D. (Figure 6).

In the previous examples, the angles of the constructed object were equal. In the present example, illustrated in Figure 6, the program constructs an object with unequal angles.

LINE-DRAWING-D
POINTS: ((-4.33 -1.56) (1.79 -1.53) (0.54 2.54) (-0.14 -0.98))
LINES: ((0 1) (0 2) (0 3) (1 2) (1 3) (2 3)))

The output object is:

OUTPUT-OBJECT-D
POINTS: ((-4.33 -1.56 0.0) (1.79 -1.53 1.0) (0.54 2.54 -0.4) (-0.14 -0.98 -4.4))
LINES: ((0 1) (0 2) (0 3) (1 2) (1 3) (2 3))

The comparison-object is:

COMPARISON-OBJECT-D
POINTS: ((-4.33 -1.56 10.0) (1.79 -1.53 30.0) (0.54 2.54 40.0) (-0.14 -0.98 50.0))

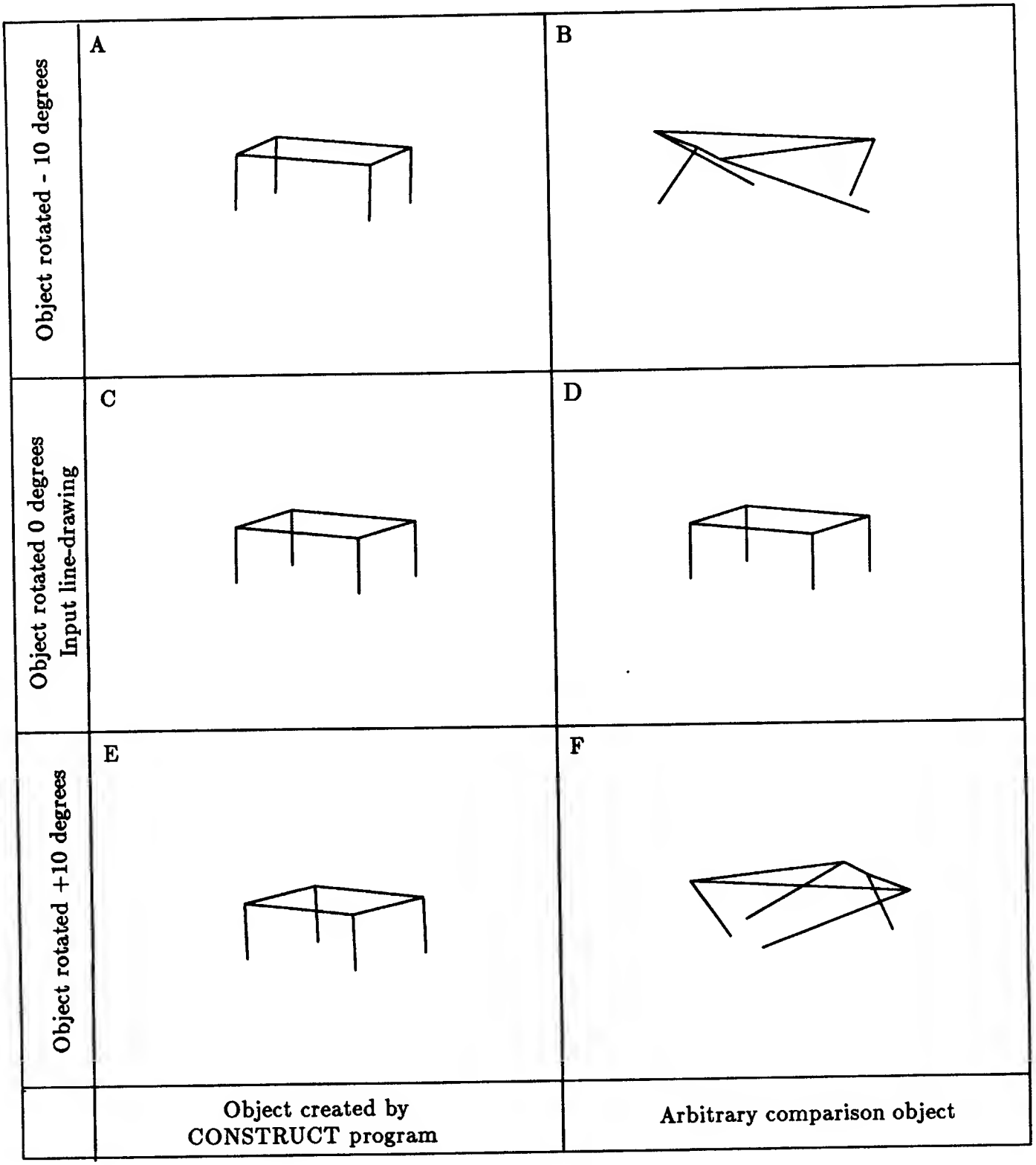

Figure 7.

Example E. See Section 3.5

LINES: ((0 1) (0 2) (0 3) (1 2) (1 3) (2 3)))

3.5 Example E. (Figure 7).

In the previous examples, the constructed objects were polyhedra (possibly with "hidden lines eliminated"). In the present example, illustrated in Figure 7, the program constructs an object which is not a polyhedron.

LINE-DRAWING-E

POINTS: ((-0.67 1.5) (3.67 1.07) (1.67 0.47) (-2.67 0.9) (-0.67 -0.47) (3.67 -0.9) (1.67 -1.5) (-2.67 -1.07))

LINES: ((0 1) (1 2) (2 3) (3 0) (0 4) (1 5) (2 6) (3 7))

The output object is:

OUTPUT-OBJECT-E

POINTS: ((-0.67 1.5 0.0) (3.67 1.07 -2.7) (1.67 0.47 -5.7) (-2.67 0.9 -3.1) (-0.67 -0.47 0.4) (3.67 -0.9 -2.3) (1.67 -1.5 -5.3) (-2.67 -1.07 -2.7))

LINES: ((0 1) (1 2) (2 3) (3 0) (0 4) (1 5) (2 6) (3 7))

The comparison-object is:

COMPARISON-OBJECT-E

POINTS: ((-0.67 1.5 20.0) (3.67 1.07 0.0) (1.67 0.47 20.0) (-2.67 0.9 0.0) (-0.67 -0.47 0.0) (3.67 -0.9 5.0) (1.67 -1.5 -10.0) (-2.67 -1.07 8.0))

LINES: ((0 1) (1 2) (2 3) (3 0) (0 4) (1 5) (2 6) (3 7))

3.6 Example F. (Figure 8).

The present example is illustrated in Figure 8. The constructed object is a truncated pyramid.

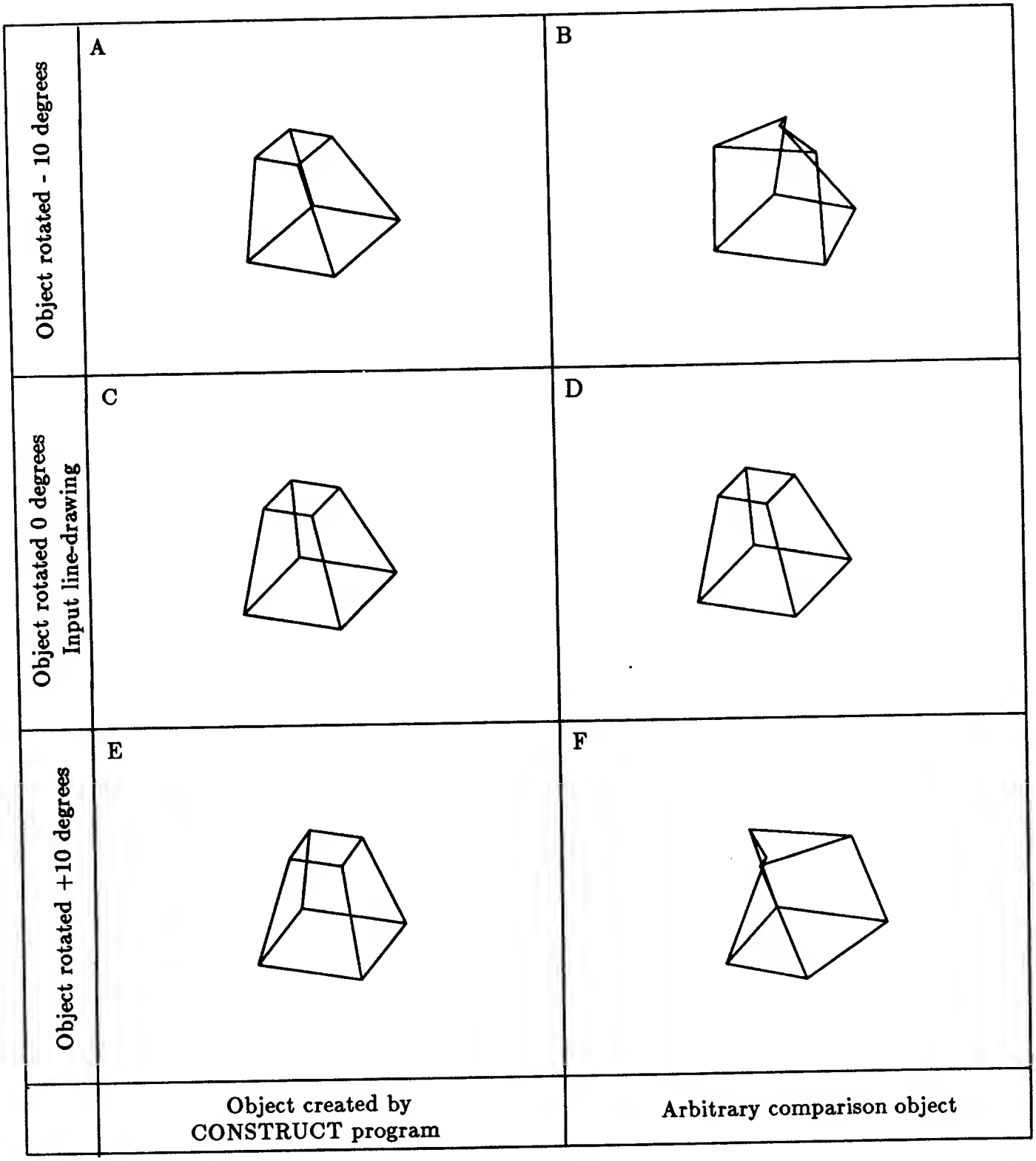

Figure 8.

Example F. See Section 3.6

LINE-DRAWING-F

POINTS: ((-2.41 -3.41) (-0.41 -1.41) (3.0 -2.0) (1.0 -4.0) (-1.65 0.35) (-0.65 1.35) (1.06 1.06) (0.06 0.06))

LINES: ((0 1) (1 2) (2 3) (3 0) (4 5) (5 6) (6 7) (7 4) (0 4) (1 5) (2 6) (3 7))

The output object is:

OUTPUT-OBJECT-F

POINTS: ((-2.41 -3.41 0.0) (-0.41 -1.41 -2.3) (3.0 -2.0 -0.5) (1.0 -4.0 1.7) (-1.65 0.35 2.3) (-0.65 1.35 0.8) (1.06 1.06 1.9) (0.06 0.06 3.3))

LINES: ((0 1) (1 2) (2 3) (3 0) (4 5) (5 6) (6 7) (7 4) (0 4) (1 5) (2 6) (3 7))

The comparison-object is:

COMPARISON-OBJECT-F

POINTS: ((-2.41 -3.41 0.0) (-0.41 -1.41 -1.0) (3.0 -2.0 2.0) (1.0 -4.0 -3.0) (-1.65 0.35 4.0) (-0.65 1.35 -5.0) (1.06 1.06 6.0) (0.06 0.06 -7.0))

LINES: ((0 1) (1 2) (2 3) (3 0) (4 5) (5 6) (6 7) (7 4) (0 4) (1 5) (2 6) (3 7))

3.7 Example G. (Figure 9).

The line-drawing used in this example (figure 9) is taken from Kanade [11]. It is the image of an "origami-world" object and appears to be a perspective. It is an interesting object in that the line-labelling schemes of Huffman [4] and Clowes [5] will reject this drawing as being the image of an "impossible object."[10]

LINE-DRAWING-G

[10] As already mentioned (footnote 2), a given drawing on a sheet of paper may correspond to several internal representations. In the present example there is a T-junction at point 4. This could be represented as three lines. Alternatively, following Roberts [2], one can use the heuristic that collinear lines at a point are merged into one line, yielding a single line at point 4. This is the approach taken here.

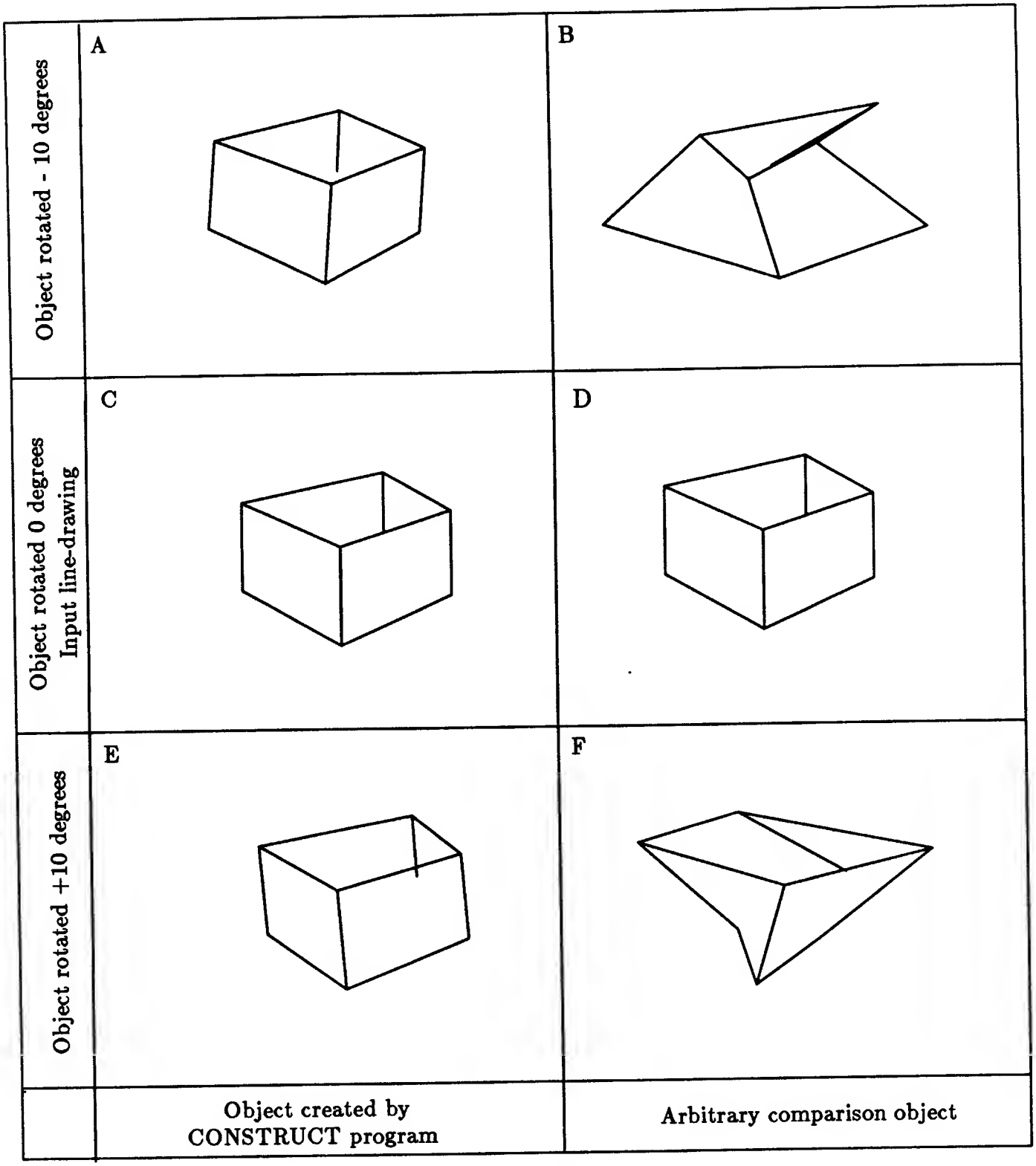

Figure 9.

Example G. See Section 3.7

POINTS: ((-2.67 -0.47) (0.8 -2.47) (0.8 1.07) (-2.67 2.67) (2.33 1.55) (4.7 -0.7) (4.7 2.33) (2.33 3.7))

LINES: ((0 3) (0 1) (1 2) (2 3) (1 5) (5 6) (2 6) (3 7) (7 4) (7 6))

The output object is:

OUTPUT-OBJECT-G

POINTS: ((-2.67 -0.47 0.0) (0.8 -2.47 -3.8) (0.8 1.07 -5.4) (-2.67 2.67 -1.5) (2.33 1.55 2.1) (4.7 -0.7 -0.9) (4.7 2.33 -2.3) (2.33 3.7 1.3))

LINES: ((0 3) (0 1) (1 2) (2 3) (1 5) (5 6) (2 6) (3 7) (7 4) (7 6))

The comparison-object is:

COMPARISON-OBJECT-G

POINTS: ((-2.67 -0.47 14.0) (0.8 -2.47 -2.0) (0.8 1.07 4.0) (-2.67 2.67 -6.0) (2.33 1.55 8.0) (4.7 -0.7 -10.0) (4.7 2.33 12.0) (2.33 3.7 -14.0))

LINES: ((0 3) (0 1) (1 2) (2 3) (1 5) (5 6) (2 6) (3 7) (7 4) (7 6))

3.8 Example H. (Figure 10).

In the preceding examples, each point in the line-drawing represented the terminus of one, two, or three lines. Thus at each point in the constructed object there were zero, one, or three angles. In the present examples there are four or five lines at each point, hence six or ten angles. The constructed object has a total of 52 angles of a variety of sizes.

The example is illustrated in Figure 10.

LINE-DRAWING-H

POINTS: ((-2.51 -0.59 0.0) (0.9 1.03 0.0) (2.51 0.59 0.0) (-0.9 -1.03 0.0) (0.93 -2.57 0.0) (-0.93 2.57 0.0))

LINES: ((1 2) (2 3) (0 1) (0 3) (4 0) (4 1) (4 2) (4 3) (5 0) (5 1) (5 2) (5 3) (0

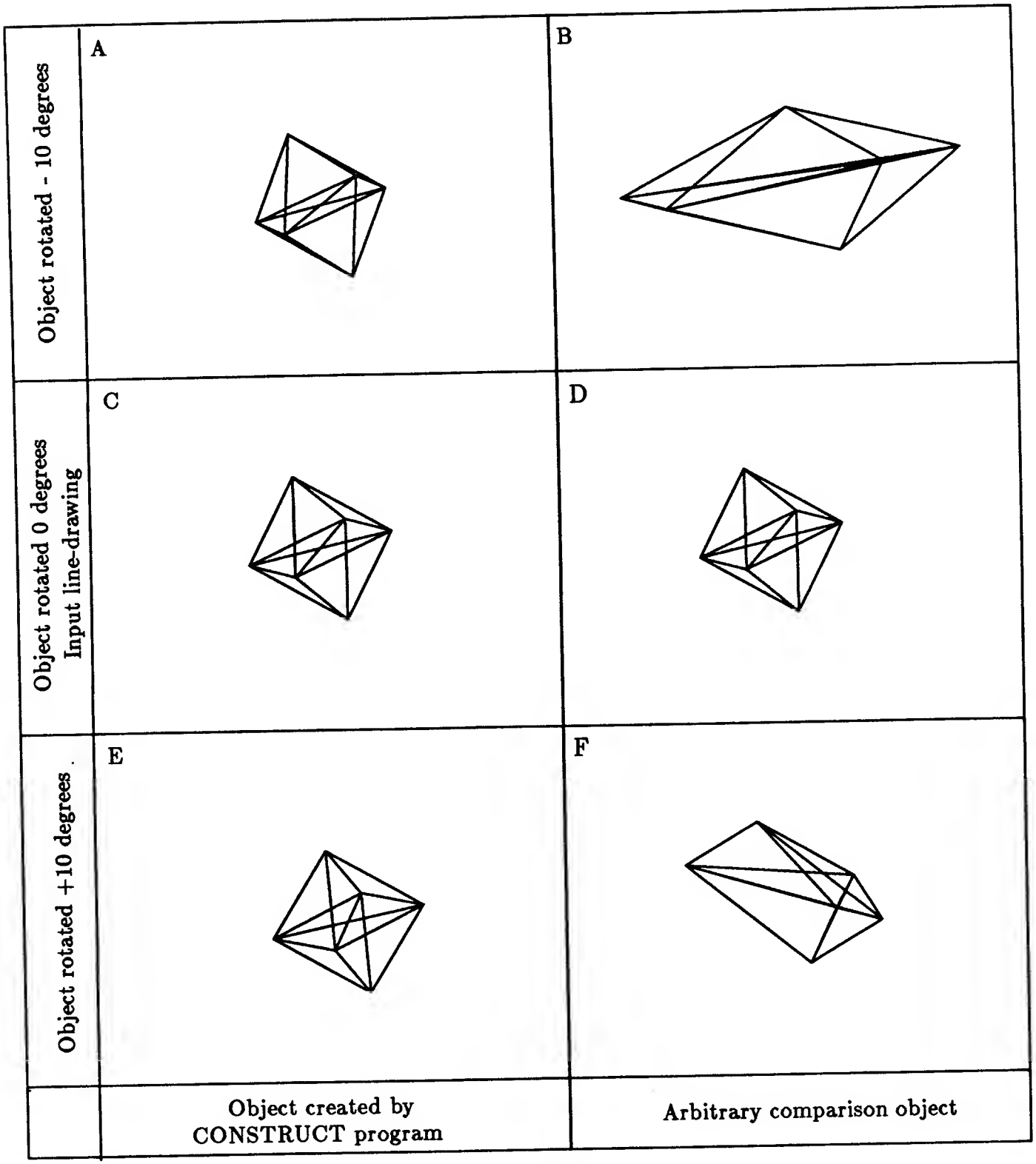

Figure 10.

Example H. See Section 3.8

2) (1 3))

The output object is:

OUTPUT-OBJECT-H

POINTS: ((-2.51 -0.59 0.0) (0.9 1.03 -1.3) (2.51 0.59 2.1) (-0.9 -1.03 3.4) (0.93 -2.57 0.1) (-0.93 2.57 2.0))

LINES: ((1 2) (2 3) (0 1) (0 3) (4 0) (4 1) (4 2) (4 3) (5 0) (5 1) (5 2) (5 3) (0 2) (1 3))

The comparison-object is:

COMPARISON-OBJECT-H

POINTS: ((-2.51 -0.59 25.0) (0.9 1.03 -25.0) (2.51 0.59 0.0) (-0.9 -1.03 25.0) (0.93 -2.57 0.0) (-0.93 2.57 0.0))

LINES: ((1 2) (2 3) (0 1) (0 3) (4 0) (4 1) (4 2) (4 3) (5 0) (5 1) (5 2) (5 3) (0 2) (1 3))

3.9 Example I. (Figure 11).

The object constructed in the present example is somewhat more complex: it has 39 angles, 21 lines, and represents a concave polyhedron with hidden lines eliminated.

The example is illustrated in Figure 11.

LINE-DRAWING-I

POINTS: ((-2.4 -1.45) (0.14 2.07) (-0.71 1.71) (-0.71 0.9) (-1.56 0.54) (-1.56 -0.28) (-2.4 -0.63) (0.78 -2.79) (3.32 -1.72) (3.32 0.73) (2.47 0.37) (2.47 -0.45) (1.63 -0.8) (1.63 -1.62) (0.78 -1.98))

LINES: ((1 2) (2 3) (3 4) (4 5) (5 6) (6 0) (7 8) (8 9) (9 10) (10 11) (11 12) (12 13) (13 14) (14 7) (0 7) (1 9) (2 10) (3 11) (4 12) (5 13) (6 14))

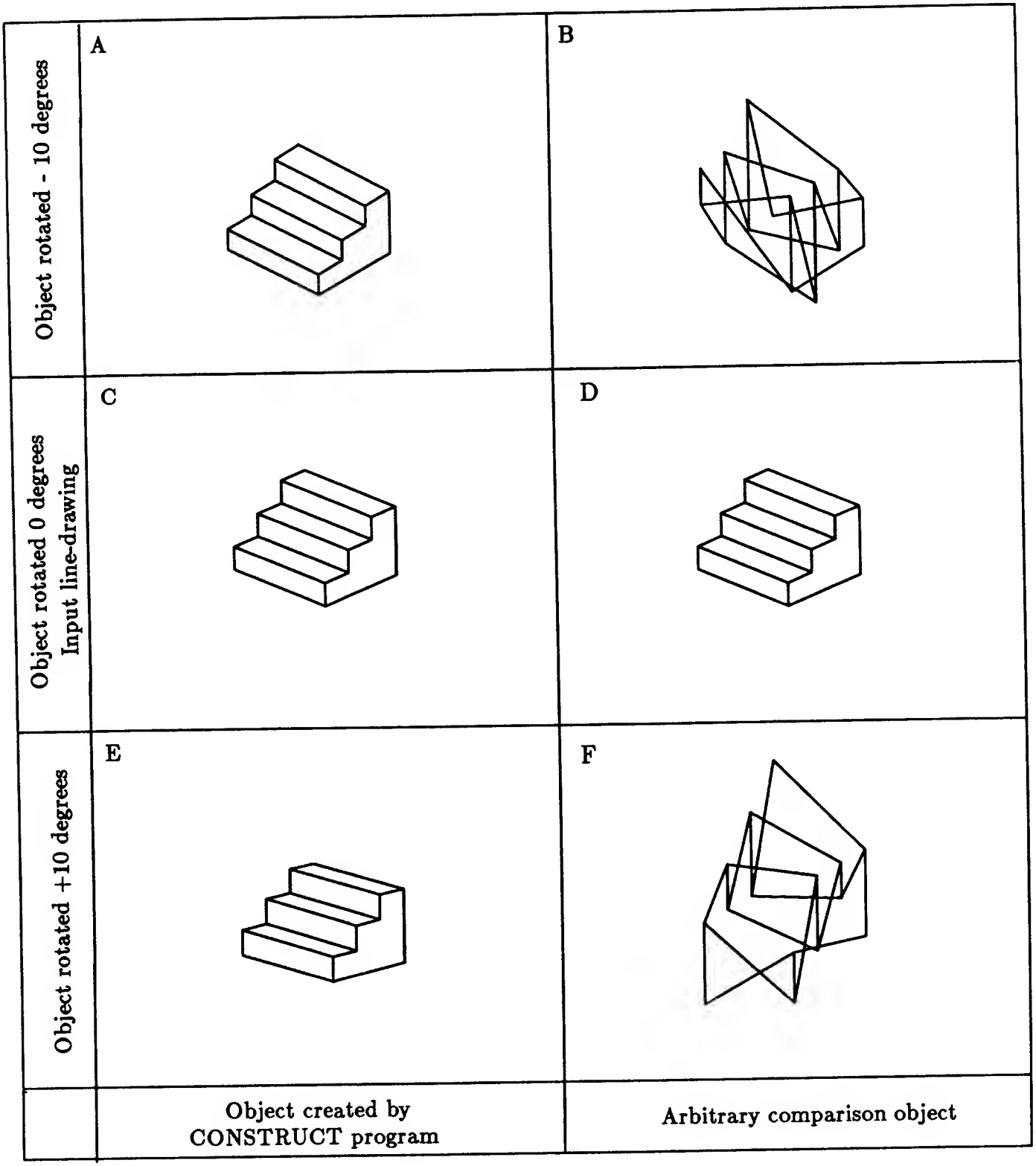

Figure 11.

Example I. See Section 3.9

The output object is:

OUTPUT-OBJECT-I

POINTS: ((-2.4 -1.45 0.0) (0.14 2.07 1.3) (-0.71 1.71 0.3) (-0.71 0.9 0.7) (-1.56 0.54 -0.2) (-1.56 -0.28 0.3) (-2.4 -0.63 -0.5) (0.78 -2.79 -2.1) (3.32 -1.72 0.3) (3.32 0.73 -0.9) (2.47 0.37 -1.8) (2.47 -0.45 -1.3) (1.63 -0.8 -2.1) (1.63 -1.62 -1.6) (0.78 -1.98 -2.5))

LINES: ((1 2) (2 3) (3 4) (4 5) (5 6) (6 0) (7 8) (8 9) (9 10) (10 11) (11 12) (12 13) (13 14) (14 7) (0 7) (1 9) (2 10) (3 11) (4 12) (5 13) (6 14))

The comparison-object is:

COMPARISON-OBJECT-I

POINTS: ((-2.4 -1.45 16.0) (0.14 2.07 -14.0) (-0.71 1.71 12.0) (-0.71 0.9 -10.0) (-1.56 0.54 8.0) (-1.56 -0.28 -6.0) (-2.4 -0.63 4.0) (0.78 -2.79 -2.0) (3.32 -1.72 1.0) (3.32 0.73 -3.0) (2.47 0.37 5.0) (2.47 -0.45 -7.0) (1.63 -0.8 9.0) (1.63 -1.62 -11.0) (0.78 -1.98 13.0))

LINES: ((1 2) (2 3) (3 4) (4 5) (5 6) (6 0) (7 8) (8 9) (9 10) (10 11) (11 12) (12 13) (13 14) (14 7) (0 7) (1 9) (2 10) (3 11) (4 12) (5 13) (6 14))

4. Related Work.

There is a large literature dealing with line-drawings (see Sugihara [1] for an excellent bibliography). This literature contains many outstanding contributions, some of which have been referred to above. It would be impossible for us to review this body of work here. It would also be inapproriate, since many of the papers differ greatly in general approach and problem definition from our own work. In the present section we wish to mention and discuss briefly those papers (known to us) that relate directly to our approach.

The historical antecedents of our approach go back to the Gestalt psychologists (Koffka, Köhler and Wertheimer). According to this school of thought, there

are global "principles of organization" that explain the nature of psychological processes, including vision. The most famous such principle is the law of Prägnanz, which holds that "psychological organization will always be as 'good' as the prevailing conditions allow." (Koffka [12]).

More recently, Attneave & Frost [13] have reinterpreted Prägnanz as "minimum complexity", and have performed psychological experiments dealing with the question of whether the perception of line-drawings of parallelepipeds could be understood in terms of a system that "minimizes diversity in angles, lengths and slopes."

More recently yet, Brady & Yuille [14] have discussed the use of "extremum principles" in establishing three-dimensional shape. These authors prove a mathematical theorem which (in our terms) can be expressed as follows:

Start with a drawing of an ellipse; consider the orthographic extension[11] of this drawing; from among the members of the extension, restrict attention to those that are planar (this is the "planarity assumption"); from among these planar objects, select those that satisfy the authors' extremum principle, which is to maximize the ratio of the area to the square of the perimeter; the resulting objects are then circles.

As a shorthand, the authors say that their extremum principle "interprets ellipses as slanted circles" (under the planarity assumption).

The authors relate this mathematical result to the vision field with the statement that "ellipses are normally perceived as slanted circles." However, this statement is unsupported, and is open to question. (It would seem at least equally plausible to say that ellipses are normally perceived as ellipses.)

Brady & Yuille also show that their extremum principle interprets a parallelogram as a slanted square and a triangle as a slanted equilateral triangle.

Also related is the work of Barnard [15], who shows that a triangle line-drawing can be mathematically "interpreted" as a slanted equilateral triangle under the assumption of equi-angularity.

To relate these workers' results to our approach, we performed the following experiments.

We input a parallelogram to the CONSTRUCT program:

---

[11] As defined in Section 2.1 above

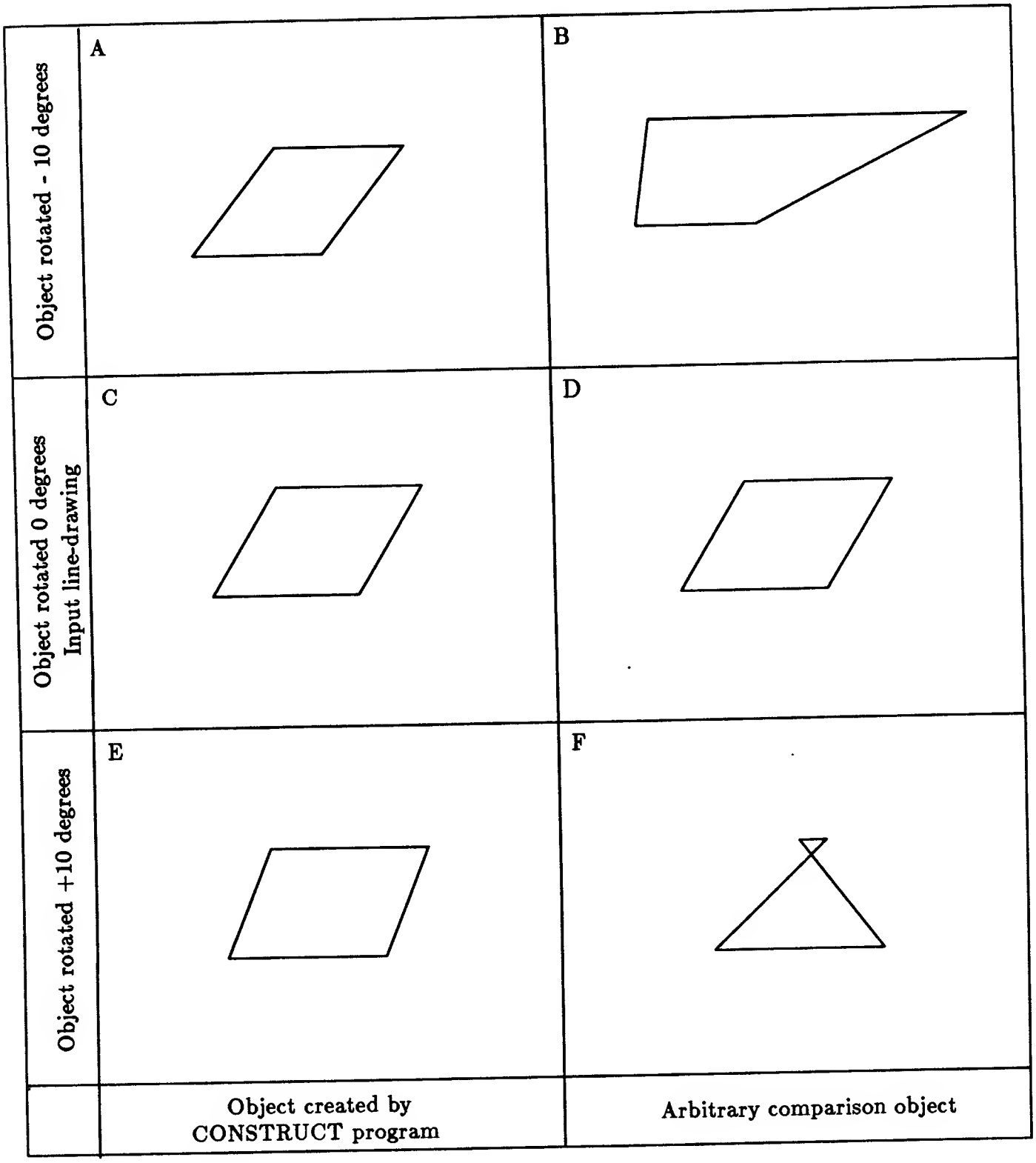

Figure 12.

Parallelogram used in experiment with CONSTRUCT

PARALLELOGRAM-LINE-DRAWING
POINTS: ((-2.75 -0.93) (2.45 -0.93) (4.75 2.93) (-0.45 2.93))
LINES: ((0 1) (1 2) (2 3) (3 0))

This line-drawing is shown in Figure 12 cell C. It has the following properties:

LINE-LENGTHS: (5.2 4.49 5.2 4.49)
ANGLES: (59.2 120.8 59.2 120.8)
SDA: 30.80

The program produced the following output:

OBJECT-FROM-PARALLELOGRAM-LINE-DRAWING
POINTS: ((-2.75 -0.93 0.0) (2.45 -0.93 2.9) (4.75 2.93 -1.2) (-0.45 2.93 -4.1))
LINES: ((0 1) (1 2) (2 3) (3 0))
ANGLES: (89.94 90.06 89.94 90.06)
SDA: 0.06
LINE-LENGTHS: (5.95 6.08 5.95 6.08)

It will be seen that OBJECT-FROM-PARALLELOGRAM-LINE-DRAWING is a slanted square (to within the accuracy of the program). Note that this experimental result was obtained *without the planarity assumption*. Note also that it would have been possible for the program to generate a non-planar object, as demonstrated by the following comparison object illustrated in the right-hand column of Figure 12.

COMPARISON-OBJECT-FOR-PARALLELOGRAM:
POINTS: ((-2.75 -0.93 0.0) (2.45 -0.93 5.0) (4.75 2.93 -25.0) (-0.45 2.93 10.0))
LINES: ((0 1)(1 2)(2 3)(3 0))
ANGLES:(38.37 50.88 8.51 29.31)
SDA: 15.46
LINE-LENGTHS: (7.21 30.33 35.38 10.96)

As a second experiment we input the following triangle into the CONSTRUCT program:

TRIANGLE-LINE-DRAWING
POINTS: ((0.0 0.0) (4.0 0.0) (0.0 4.0))
LINES: ((0 1)(1 2)(0 2))
This line-drawing is a right triangle.

The output was the following:
OBJECT-FROM-TRIANGLE-LINE-DRAWING:
POINTS: ((0.0 0.0 0.0) (4.0 0.0 4.0) (0.0 4.0 4.0))
LINES: ((0 1)(1 2)(0 2))
ANGLES: (60.0 60.0 60.0)
SDA: 0.0
LINE-LENGTHS: (5.66 5.66 5.66)

It will be seen that OBJECT-FROM-TRIANGLE-LINE-DRAWING is a slanted equilateral triangle.

Thus in these experiments the CONSTRUCT program, and hence the MSDA principle, generated the same output as the principles of Brady& Yuille and of Barnard.

5. Why does MSDA work?

None of the above sheds light, however, on the question of *why* MSDA works as well as it does. We can speculate that by minimizing the standard-deviation of angles in the constructed object we are minimizing some quantity of fundamental importance to the human vision system: perhaps the number of bits used to represent the object. However, there is no evidence to support such speculation. For the moment, the question remains open. Further investigations into the nature of human vision may provide the answer.

Acknowledgments.

I would like to thank Rodney Brooks, with whom I had many discussions, all of them illuminating; Boris Katz, who managed, through magic, to eliminate the more egregious flaws from the program; and Tomás Lozano-Pérez, who read an early draft and provided insightful comments.

References.

[1] K. Sugihara, *Machine Interpretation of Line Drawings.* Cambridge, Mass.: MIT Press (1986).

[2] L. G. Roberts, "Machine perception of three-dimensional solids." In Tippett et al. (Eds.), *Optical and Electro-Optical Information Processing.* Cambridge, Mass.: MIT Press (1965).

[3] E. Charniak & D. McDermott, *Introduction to Artificial Intelligence,* Reading, Mass.: Addison-Wesley (1985).

[4] D.A. Huffman, "Impossible objects as nonsense sentences." In B. Meltzer & D. Michie (Eds.), Machine Intell. **6** (1971).

[5] M. B. Clowes, "On seeing things." Artif. Intell. **2** 79-116 (1971).

[6] A. K. Mackworth, "Interpreting pictures of polyhedral scenes," Artif. Intell., 4, 121-137 (1973).

[7] T. Kanade, "A theory of origami world." Artif. Intell. **13** 279-311 (1980).

[8] D.H. Ballard & C. M. Brown, *Computer Vision.* Englewood Cliffs, N.J.: Prentice-Hall (1982)

[9] D. G. Lowe, "Three-dimensional object recognition from single two-dimensional images," Artif. Intell., **31**, 355-395 (1987).

[10] P. H. Winston, *Artificial Intelligence.* Second edition, Reading, Mass.: Addison-Wesley (1984).

[11] T. Kanade, "Recovery of the 3-D shape of an object from a single view," Artif. Intell. **17**, 409-460 (1981).

[12] K. Koffka, *Principles of Gestalt Psychology.* N.Y.: Harcout Brace (1935).

[13] F. Attneave & R. Frost, "The determination of perceived tridimensional orien-

tation by minimum criteria," Perception and Psych. **6** 391-396 (1969)

[14] M. Brady & A. Yuille, "An extremum principle for shape from contour." AI Memo 711, MIT Artificial Intelligence Lab. (June 1983).

[15] S. T. Barnard, "Interpreting perspective images," Artif. Intell. **21**, 435-462 (1983).

Appendix: Supplementary Information for Examples.

In this appendix we give the angles, the SDA, and the line-lengths for each object constructed in the examples and each comparison object.

Example A. (Figure 3).

OUTPUT-OBJECT-A

ANGLES: (89.83 90.63 90.0 89.83 89.37 90.0 89.37 90.17 90.0 90.17 90.63 90.0 90.17 89.37 90.0 90.17 90.63 90.0 90.63 89.83 90.0 89.83 89.37 90.0)

SDA: 0.38

LINE-LENGTHS: (4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.08 4.08 4.08 4.08)

COMPARISON-OBJECT-A

ANGLES: (147.35 106.08 62.68 147.35 133.63 67.91 172.75 140.62 36.39 140.62 146.72 39.38 15.63 25.18 14.3 15.63 5.73 15.42 7.69 19.78 12.04 19.78 21.87 13.09)

SDA: 58.05

LINE-LENGTHS: (4.0 7.19 10.58 7.19 18.33 22.35 26.23 22.35 8.27 8.27 8.27 8.27)

Example B. (Figure 4).

OUTPUT-OBJECT-B

ANGLES: (90.23 90.8 89.77 89.2 88.97 89.77 90.8 91.03 88.97 89.2 90.23 91.03 90.23 90.8 88.97)

SDA: 0.77

LINE-LENGTHS: (4.05 6.04 4.05 6.04 4.05 6.04 3.8 3.8 3.8)

COMPARISON-OBJECT-B

ANGLES: (131.45 160.73 116.74 135.97 65.04 20.61 33.38 20.28 24.22 10.26 28.6 18.19 28.6 23.94 21.09)

SDA: 50.43

LINE-LENGTHS: (6.93 6.99 14.42 18.89 22.27 18.89 6.43 8.33 8.33)

Example C. (Figure 5).

OUTPUT-OBJECT-C

ANGLES:(60.2 60.2 60.0 60.2 60.2 60.0 60.2 60.2 60.0 59.54 59.54 59.54)

SDA: 0.27

LINE-LENGTHS: (4.0 4.0 4.03 4.0 4.03 4.03)

COMPARISON-OBJECT-C

ANGLES: (5.13 8.51 9.94 16.46 162.75 150.93 146.3 160.05 19.09 17.25 14.98 8.89)

SDA: 67.38

LINE-LENGTHS: (20.4 30.27 40.07 10.77 20.13 10.26)

Example D. (Figure 6).

OUTPUT-OBJECT-D

ANGLES: (50.95 55.88 41.75 61.84 61.18 71.28 70.85 61.84 66.98 47.31 67.23 62.94)

SDA: 8.92

LINE-LENGTHS: (6.2 6.38 6.1 4.48 5.76 5.37)

COMPARISON-OBJECT-D

ANGLES: (7.69 11.19 10.89 20.45 157.37 147.89 138.5 150.0 21.25 20.93 22.33 11.48)

SDA: 62.82

LINE-LENGTHS: (20.91 30.67 40.22 10.87 20.1 10.62)

Example E. (Figure 7).

OUTPUT-OBJECT-E

ANGLES: (90.4 91.26 90.11 90.11 88.74 89.08 88.97 89.89 91.83 89.6 91.03 88.97)

SDA: 0.95

LINE-LENGTHS: (5.12 3.66 5.07 3.74 2.01 2.01 2.01 2.01)

COMPARISON-OBJECT-E

ANGLES: (6.78 12.84 18.01 20.61 25.71 7.25 13.09 8.11 18.01 16.46 17.45 7.25)

SDA: 5.84

LINE-LENGTHS: (20.47 20.11 20.47 20.11 20.1 5.37 30.06 8.24)

Example F. (Figure 8).

OUTPUT-OBJECT-F

ANGLES: (75.52 76.64 82.88 78.76 82.18 96.14 78.76 76.05 83.05 80.44 80.21 97.99 106.26 96.78 91.2 104.95 104.24 86.79 105.01 100.14 91.78 97.7 95.68 90.23)

SDA: 10.11

LINE-LENGTHS: (3.65 3.9 3.58 3.86 2.06 2.05 1.99 2.0 4.48 4.16 4.35 4.46)

COMPARISON-OBJECT-F

ANGLES: (117.13 72.24 50.95 130.24 128.39 101.13 135.81 98.34 38.28 104.0 114.52 69.21 44.85 51.54 8.11 33.49 29.66 15.85 41.15 48.55 7.25 50.13 49.68 13.59)

SDA: 40.38

LINE-LENGTHS: (3.0 4.58 5.74 4.58 9.11 11.14 13.08 11.14 5.55 4.87 5.4 5.78)

Example G. (figure 9)

OUTPUT-OBJECT-G

ANGLES: (91.72 85.35 87.94 87.31 88.85 88.57 88.45 81.37 87.25 92.46 94.3 99.03 86.62 91.72 89.66 90.8 90.11)

SDA: 3.76

LINE-LENGTHS: (3.48 5.52 3.88 5.46 5.17 3.34 5.14 5.82 2.29 4.52)

COMPARISON-OBJECT-G

ANGLES: (19.95 131.3 157.22 26.23 161.99 146.93 43.2 129.27 148.21 18.92 32.11 23.51 12.04 25.71 5.73 37.25 31.79)

SDA: 58.72

LINE-LENGTHS:(20.24 16.49 6.97 10.71 9.07 22.21 8.99 9.49 22.1 26.14)

Example H (Figure 10).

OUTPUT-OBJECT-H

ANGLES: (45.49 46.61 45.73 42.86 92.12 58.87 60.79 63.19 57.93 88.57 45.17 48.39 44.19 47.23 93.55 60.33 60.66 60.86 63.96 91.43 46.61 45.49 42.86 45.73 92.12 57.93 63.19 60.79 58.87 88.57 48.39 45.17 47.23 44.19 93.55 63.96 60.86 60.66 60.33 91.43 58.87 86.45 56.15 57.18 87.88 61.18 61.18 86.45 57.18 56.15 87.88 58.87)

SDA: 16.39

LINE-LENGTHS: (3.79 3.99 3.99 3.79 3.98 3.86 4.05 4.07 4.05 4.07 3.98 3.86 5.57 5.44)

COMPARISON-OBJECT-H

ANGLES: (8.89 8.11 79.75 7.69 12.31 88.45 5.13 81.26 7.25 86.73 6.28 6.28 2.56 5.73 12.58 5.13 9.25 7.69 8.11 7.69 81.49 82.59 3.62 164.8 93.27 85.12 86.27 83.22 89.26 168.24 6.28 6.28 91.38 5.73 12.58 87.76 8.89 94.99 8.11 96.49 88.74 167.42 3.62 82.7 89.48 165.02 85.87 167.42 3.62 84.61 89.54 169.42)

SDA: 54.47

LINE-LENGTHS: (25.06 25.28 50.14 1.68 25.31 25.26 3.53 25.11 25.25 25.11 3.98 25.26 25.53 50.07)

Example I. (Figure 11).

OUTPUT-OBJECT-I

ANGLES: (90.69 90.17 85.99 88.8 85.01 94.65 94.7 86.67 87.76 84.26 82.65 98.34 92.24 84.72 88.57 82.7 84.72 85.99 96.26 91.43 88.57 87.42 93.33 93.33 90.69 93.61 82.65 82.7 88.57 84.72 92.24 98.34 84.72 84.26 87.76 82.65 94.7 94.65 86.67)

SDA: 4.57

LINE-LENGTHS: (1.36 0.91 1.29 0.96 1.22 0.96 3.66 2.73 1.29 0.96 1.22 0.96 1.29 0.91 4.04 4.1 4.04 3.99 3.94 3.94 3.99)

COMPARISON-OBJECT-I

ANGLES: (12.84 18.92 25.44 24.91 3.62 51.1 49.91 4.44 104.83 104.06 5.13 148.87 146.51 7.25 19.78 155.09 174.87 9.94 50.73 41.5 60.66 157.52 151.04 29.42 154.96 157.82 8.89 128.61 129.35 5.73 74.75 76.17 4.44 33.18 33.8 3.62 22.33 22.63 4.44)

SDA: 57.95

LINE-LENGTHS: (26.02 22.02 18.02 14.02 10.04 12.03 4.08 4.69 8.05 12.03 16.03 20.02 24.02 15.02 18.33 11.53 7.81 4.58 3.6 6.08 9.64)

CS-TR Scanning Project
Document Control Form

Date : 1/19/95

Report # AIM-1136

Each of the following should be identified by a checkmark:
Originating Department:

☒ Artificial Intellegence Laboratory (AI)
☐ Laboratory for Computer Science (LCS)

Document Type:

☐ Technical Report (TR) ☒ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 40 (46-IMAGES)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:
☒ Single-sided or
☐ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☒ Laser Print
☐ inkJet Printer ☐ Unknown ☐ Other:_____

Check each if included with document:

☒ DOD Form 2 (PGS) ☐ Funding Agent Form ☐ Cover Page
☐ Spine ☐ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages (by page number):_____

Photographs/Tonal Material (by page number):_____

Other (note description/page number):

Description :	Page Number:
IMAGE MAP	(1) UNNUMBERED TITLE PAGE
	(2) UNNUMBERED PAGE 1
	(3-40) PAGES #'ED 2-39
	(41) SCANCONTROL
	(42-44) TRCT'S
	(45-46) DOD'S

Scanning Agent Signoff:
Date Received: 1/19/95 Date Scanned: 1/24/95 Date Returned: 1/26/95

Scanning Agent Signature: Michael W. Cook

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AIM 1136	2. GOVT ACCESSION NO.: AD-A215058
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): Computer Perception of Three-Dimensional Objects	5. TYPE OF REPORT & PERIOD COVERED: memorandum
	6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): Thomas Marill	8. CONTRACT OR GRANT NUMBER(s): N00014-85-K-0124
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Artificial Intelligence Laboratory, 545 Technology Square, Cambridge, MA 02139	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
11. CONTROLLING OFFICE NAME AND ADDRESS: Advanced Research Projects Agency, 1400 Wilson Blvd., Arlington, VA 22209	12. REPORT DATE: August 1989 13. NUMBER OF PAGES: 40
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Office of Naval Research, Information Systems, Arlington, VA 22217	15. SECURITY CLASS. (of this report): UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

Distribution is unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer vision
three-dimensional
minimization
line drawings
perception
psychological vision

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

We first pose the following problem: To develop a program which takes line drawings as input and constructs three-dimensional objects as output; such that the output objects are the same as the ones we see when we look at the input line-drawing. We then introduce the principle of minimum standard-deviation of angles (MSDA) and discuss a program based on MSDA. We present the results of testing this program with a variety of line-drawings and show that the program constitutes a solution to the stated problem over the range of line-drawings tested. (cont on reverse)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

1. REPORT NUMBER: AIM 1130

2. GOVT ACCESSION NO: AD-A150 052

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle): Computer Perception of Three-Dimensional Objects

5. TYPE OF REPORT & PERIOD COVERED: memorandum

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s): Tomas Merill

8. CONTRACT OR GRANT NUMBER(s): N00014-85-K-0124

9. PERFORMING ORGANIZATION NAME AND ADDRESS: Artificial Intelligence Laboratory, 545 Technology Square, Cambridge, MA 02139

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS: Advanced Research Projects Agency, 1400 Wilson Blvd., Arlington, VA 22209

12. REPORT DATE: August 1985

13. NUMBER OF PAGES: 40

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Office of Naval Research, Information Systems, Arlington, VA 22217

15. SECURITY CLASS. (of this report): UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report): Distribution is unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES: None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number): Computer vision; three-dimensional; minimization; line drawings; perception; psychological vision

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

We first pose the following problem: To develop a program which takes line drawings as input and constructs three-dimensional objects as output; such that the output objects are the same as the ones we see when we look at the input line-drawing. We then introduce the principle of minimum standard-deviation of angles (MSDA) and discuss a program based on MSDA. We present the results of testing this program with a variety of line-drawings and show that the program constitutes a solution to the stated problem over the range of line-drawings tested. (cont on reverse)

DD FORM 1473, 1 JAN 73 — EDITION OF 1 NOV 65 IS OBSOLETE — S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

UNCLASSIFIED

Finally, we relate this work to its intentional antecedents in the psychological and computer-vision literature.